INFORMATION AND CONTROL 26, 351–368 (1974)

A Logic Hazard Detection and Elimination Method*

LEVENTE L. MÁTÉ

Computer and Automation Institute of the Hungarian Academy of Sciences, Hungary

SANTANU DAS†

Paul H. Henson Research Center, North Electric Co., Delaware, Ohio

AND

HENRY Y. H. CHUANG†

Department of Computer Science, University of Pittsburgh, Pittsburgh, Pennsylvania 15213

This paper is concerned with the detection and elimination of static logic hazards due to single and multiple input changes. The method involves only simple manipulation of the Boolean expression corresponding to the logic network. It not only detects the existence of hazards and determines the variable sets with respect to which the hazards exist, but also, for each of these sets, pin-points the subcubes within which each input transition involving exactly the changes of all variables in the set produces the hazardous output. Further, it can determine the terms the absence of which from the expression cause the hazards. Thus, the hazard can be eliminated by including these missing terms in the expression. The theoretical foundation of the method is given first followed by an example illustrating the procedure and its effectiveness. The procedure is formulated into two different algorithms, one for saving time and the other for saving memory. They have been implemented in FORTRAN. A related result on prime implicant generation is also included.

1. INTRODUCTION

Spurious pulses occurring on the output of combinational switching circuits during an input transition are called combinational hazards. The combinational hazards [Unger, 1969; Bredeson *et al.*, 1972; Yoeli *et al.*, 1964; Eichelberger, 1965; Huffman, 1957; McClusky, 1962; McGhee, 1969; Das

* Partially supported by the National Science Foundation under Grant GJ-35569.

† S. Das and H. Y. H. Chuang were with Computer Systems Laboratory, Washington University, St. Louis.

and Chuang, 1972; Rey, 1974; Máté *et al.*, 1973] can be classified with respect to

(i) whether the number of the variables changing during the transition is one (single variable) or more (multiple variable),

(ii) whether the steady values of the output before and after the transition are identical (static) or different (dynamic),

(iii) whether the value of the function for each cell within the smallest subcube which includes the two input states before and after the transition is the same (logic) or not (function).

This paper is concerned with the detection and elimination of *static logic* hazards due to both *single* and *multiple* input changes. This problem is not new and there have been a number of attempts [Unger, 1969; Bredeson *et al.*, 1972; Yoeli *et al.*, 1964; Eichelberger, 1965; Huffman, 1957; McClusky, 1962; McGhee, 1969] in recent years in finding solution to the problem. Most of the previous works require examination of the existence of hazard for every input transition, some of them even requiring binary to ternary conversion [Yoeli, *et al.*, 1965; Eichelberger, 1965]. The most significant result in this area is due to Eichelberger [Eichelberger, 1965] who first proved that the realization of a Boolean expression would be hazard-free if and only if the expression includes all the prime implicants (implicates) of the function. On the basis of this result, it might appear that hazard detection can be easily effected by generating all the prime implicants (implicates) of the function from the given expression and then noting which are absent in the expression. But in reality, the problem is not that simple. As, for instance, in Figure 1, we see that the omission of the prime implicant $x\bar{y}$ gives rise to two variable hazards for transitions between the cells 5 and 12 as well as the cells 4 and 13. But in addition to these, there is a possibility of a single variable hazard between the cells 4 and 5. The generation of all the prime implicants and then comparing with the given expression would detect the omission of the prime implicant $x\bar{y}$. This at once locates the two variable hazards already mentioned but no information about the second hazard, namely the single-variable hazard between the cells 4 and 5, emanates from this exercise. When the number of variables involved in the expression is large, many such hazards will not be detected. Of course, if we were interested only in the elimination of the hazards, the information about these undetected hazards is not at all necessary. Unfortunately, in practice, due to "don't care" situations, we might be interested in removing *only some of the hazards* as the transitions giving rise to the other hazards could be just "don't care" situations. Thus, this simple approach based on the generation of all the prime implicants is virtually

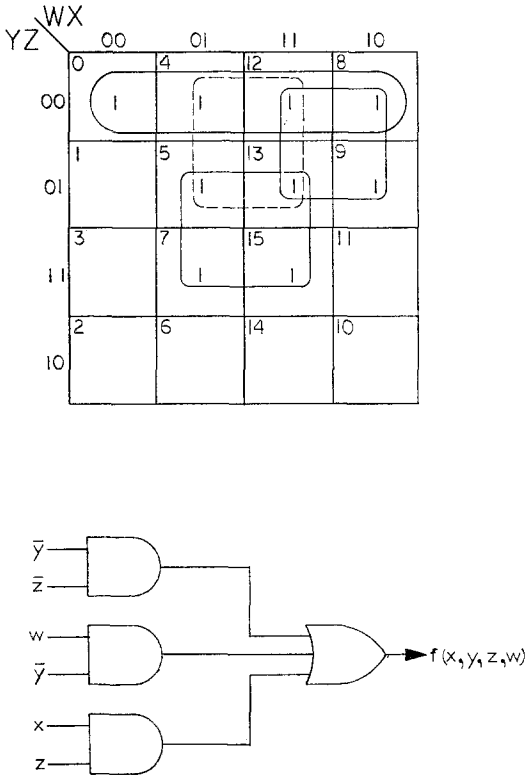

FIG. 1. Karnaugh map and logic realization of the expression $\bar{y}\bar{z} + w\bar{y} + xz$.

useless when we are interested in knowing the exact locations of all possible hazards. Moreover, the generation of prime implicants [Nelson, 1957; Slage *et al.*, 1970; Das and Khabra, 1972; McClusky, 1956; Su, 1969; Roth, 1958; Tison, 1967] of a function from a given arbitrary expression is not as simple as it might appear on the face of it. Motivated by all these considerations, we have devised an algebraic approach, which is very direct and involves only simple manipulation of the Boolean expression, corresponding to the logic network. The method is also very powerful. It not only detects the existence of hazards and determines the variable sets with respect to which the hazards exist, but also, for each of these sets, pin-points the subcubes within which each input transition involving exactly the changes of all variables in the set produces the hazardous output. Further, it can determine the terms whose absence from the expression cause the hazards. Thus, the

hazards can be eliminated by including these missing terms in the expression. Two different algorithms have been developed from one basic idea and both of them have been implemented in FORTRAN [Máté *et al.*, 1973]. The first algorithm is intended to save computing time, while the second is for saving memory space. The second one has also the flexibility that instead of detecting the hazards with respect to all the variables, we can specify the variables we are interested in. As a result, the hazards with respect to only these variables will be detected.

2. Theoretical Foundations

Terms with complementary literals play a key role in our hazard detection and elimination technique. The first definition characterizes such terms:

Definition 1. A product containing M variables ($M > 0$) both complemented and uncomplemented is called an *unstable term*, u_M. The part of u_M which includes only those variables appearing both complemented and uncomplemented is called the *unstable part*, U_M, of u_M. The other part of u_M is called its *stable part*, S_M. Thus $u_M = U_M S_M$. When $u_M = U_M$, $S_M = 1$.

Definition 2 introduces two special kinds of subcubes which are closely related with unstable terms.

Definition 2. A subcube for which the variables of U_M are not assigned is called an *M-subcube* of U_M. Those M-subcubes of U_M for which $S_M = 1$ are called the acceptable subcubes of u_M. Here M refers to the dimensionality of the unstable part as well as the subcube.

There are some input transitions, related to an unstable term, in which we are particularly interested.

Definition 3. A transition between any two input states which represent a diagonal of an M-subcube of U_M is called a *bridge transition* of U_M, and the transition is said to span this M-subcube.

To illustrate the concepts of the previous definitions, an example is presented. For a function of four variables, x, y, z, w, the unstable term $u_2 = \bar{x}y\bar{y}w\bar{w}$ consists of the unstable part $U_2 = y\bar{y}w\bar{w}$ and the stable part $S_2 = \bar{x}$. The four 2-subcubes of U_2 are 0-0-, 0-1-, 1-1-, 1-0-. Since S_2 is not identically equal to 1, not all the 2-subcubes of U_2 are acceptable subcubes of u_2. The two acceptable subcubes of u_2 are 0-0-, 0-1-. $I_1 \rightarrow I_2$ and $I_2 \rightarrow I_1$,

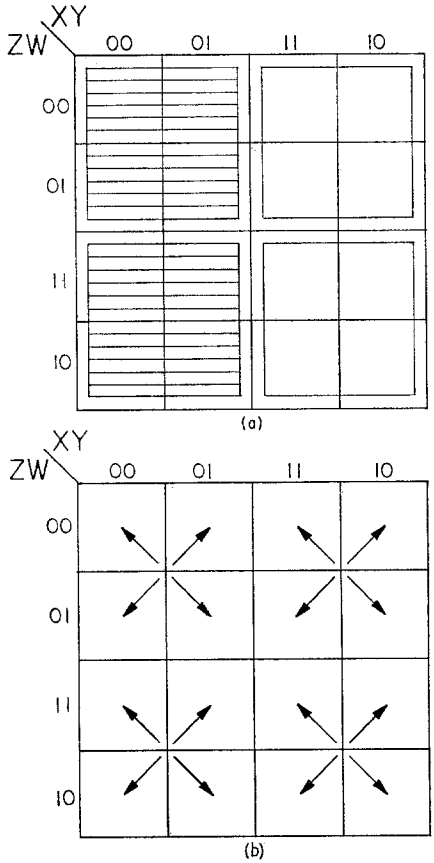

FIG. 2. (a) The 2-subcubes of $U_2 = y\bar{y}w\bar{w}$ (framed) and the acceptable subcubes of $u_2 = \bar{x}y\bar{y}w\bar{w}$ (striped). (b) The bridge transitions of $U_2 = y\bar{y}w\bar{w}$.

where $I_1 = 0100$ and $I_2 = 0001$ are two of the bridge transitions of U_2. Figure 2 shows the relevant subcubes and transitions.

Next, we analyze the role which a single unstable term plays in logic 0 hazards.

LEMMA 1. *Given two input states I_1 and I_2 differing in the value of M variables. An unstable term u_M can have a* 0-1-0 *sequence of values during the transition from I_1 to I_2 iff the M-subcube spanned by this transition is an acceptable subcube of u_M.*

Proof. Obviously the stable values of u_M before and after the transition

are 0. If the M-subcube spanned by the transition from I_1 to I_2 is an acceptable subcube of u_M, then during the transition the stable part of u_M remains identically 1, while all the variables involved in the unstable part change simultaneously. Due to stray delays it could happen that for a short moment every changing variable and its complement assume 1-value simultaneously resulting in a momentary 1-value for u_M. Thus, the "if" part of the lemma is proved. If the M-subcube spanned by this transition is not an acceptable subcube of u_M but still is an M-subcube of its unstable part, then the stable part of u_M has to be identically 0 during the transition. If the M-subcube is not an M-subcube of the unstable part of u_M, then at least one of the variables in the unstable part does not change during the transition. In neither case can the 0-1-0 sequence occur. Thus, the "only if" part of the lemma is also proved. Q.E.D.

Now that we have established the relation between a logic 0 hazard and an unstable term, we go on to our first theorem which gives the necessary and sufficient conditions for the existence of logic 0 hazard in the realization of a Boolean expression based on this relationship.

THEOREM 1. *A sum-of-products expression E, free of N-variable logic 0 hazard ($N < M$), has an M-variable logic 0 hazard within an M-subcube iff*

(1) *there is an unstable term $u_M = U_M S_M$ in E;*
(2) *the value of E is 0 for each cell within the M-subcube;*
(3) *the M-subcube is an acceptable subcube of u_M.*

Proof. The "if" part of the theorem follows from the proof of Lemma 1. The existence of an unstable term u_K for some K is necessary since no sum of stable terms can cause logic 0 hazard. If $K < M$, then u_K cannot produce M-variable logic hazard in an M-subcube since no M-subcube can contain bridge transition of U_K. If $K > M$ and u_K produces an M-variable logic 0 hazard, then u_K also causes a K variable logic 0 hazard in a K-subcube contained by the M-subcube and this contradicts the given hypothesis. Thus $K = M$, i.e., condition (1) is necessary. The necessity of condition (2) and (3) follow from the definition of logic hazard and from Lemma 1, respectively. Thus the "only if" part of the theorem is established. Q.E.D.

It can be seen that this theorem shows not only how to detect the existence of logic 0 hazard but also how to pin-point the subcubes within which any bridge-transition produces the hazardous output.

By now it is clear that Theorem 1 is based on unstable terms in sum-of-products expressions. But it is very likely that the logic designer will delete

the unstable terms from his expressions during the transformation and simplification process, thus eliminating the logic 0 hazard. However, this does not diminish the usefulness of the theorem. It can be conveniently used to detect logic 0 hazards in product-of-sums expressions or in mixed expressions. These expressions can be transformed to sum-of-products form through logic-hazard preserving transformations and the resulting sum-of-products expressions can be examined using Theorem 1 to detect possible logic 0 hazards. Unger [1969] has shown that the application of associative, distributive and DeMorgan laws are static hazard preserving. It is possible that the sum-of-products expression resulting from the application of these transformations will contain large number of terms. The next two lemmas are concerned with some additional hazard-preserving transformations which could be used to simplfy the sum-of-products expression.

LEMMA 2. *The following transformations on a sum-of-products expression will not affect its logic* 0 *hazard behavior*:

(1) *any simplification involving only stable terms and resulting in sum-of-products form*;

(2) *collecting terms with common unstable part, factoring out the unstable part, and then simplifying the resulting sum of the stable parts.*

Proof. Since neither transformation changes the conditions of Theorem 1, the resulting expression preserves the logic 0 hazard behavior of the original expression. Q.E.D.

LEMMA 3. *Given a sum-of-products expression which may or may not contain unstable terms. Application of the absorption law to the expression is logic hazard preserving.*

Proof. (1) With respect to logic 1 hazards, the statement of the lemma follows from the fact that no application of the absorption law can change the prime implicants existing in the expression.

(2) For logic 0 hazards we consider the following cases:

(i) A stable term absorbs another stable term. The proof for this case follows directly from Lemma 2.

(ii) A stable term absorbs an unstable term. The stable term generates at least one "1" within each acceptable subcube of the unstable term. Therefore with respect to this unstable term, condition (2) of Theorem 1

is never satisfied. Thus, its omission will not change the logic 0 hazard behavior of the expression.

(iii) An unstable term absorbs another unstable term. Whenever the term which can be absorbed emits a spurious 0-1-0 pulse, the term which can absorb it will also emit such a pulse.

(iv) An unstable term cannot absorb any stable term.

Hence the lemma is proved. Q.E.D.

Theoretically, we have by now sufficient tools for detecting logic 0 hazards of any expression. However, some practical difficulties arise when one attempts to use Theorem 1. In order to make our hazard detection techniques more practical, we define a new operation called $\underline{D}$-operation (deletion operation).

DEFINITION 4. The operation $\underline{D}(E/U_M)$ on an expression E with respect to an unstable part U_M consists of the following two steps:

(i) delete all the unstable terms from E (Should all terms be deleted, the result would be 0),

(ii) delete all the literals of U_M from the remaining expression and the result is the *function* represented by the final expression (Should any terms be completely deleted, the result would be 1).

For instance, if $E = \bar{x}\bar{y}z + \bar{x}\bar{y}w + xz\bar{w} + \bar{y}z\bar{w} + y\bar{y}z + x\bar{x}w\bar{w}$, $U_1 = y\bar{y}z$, then $\underline{D}(E/U_1) = \bar{x}\cancel{\bar{y}}z + \bar{x}\cancel{\bar{y}}w + xz\bar{w} + \cancel{y}z\bar{w} + \cancel{y\bar{y}z} + \cancel{x\bar{x}w\bar{w}} = \bar{x}z + \bar{x}w + z\bar{w}$. The following lemma shows the implication of the $\underline{D}$-operation.

LEMMA 4. *Given a sum-of-products expression E and unstable part U_M. The function $\underline{D}(E/U_M)$ covers an M-subcube of U_M iff the M-subcube contains at least one 1 of E.*

Proof. The function $\underline{D}(E/U_M)$ does not contain any variable of U_M. Therefore, it can not have different values within any M-subcube of U_M. Moreover, deleting a variable from a sum-of-products expression can not change the value of the function from 1 to 0. Hence, each M-subcube of U_M which contains at least one 1 of E is covered by $\underline{D}(E/U_M)$. Any M-subcube of U_M containing only 0's of E can not be covered by $\underline{D}(E/U_M)$, since the values of E in the M-subcube do not depend on the variables of U_M. Q.E.D.

Figure 3 illustrates how $\underline{D}(E/U_1)$ covers 1-subcubes of U_1. Now we are ready to give a more practical form of Theorem 1.

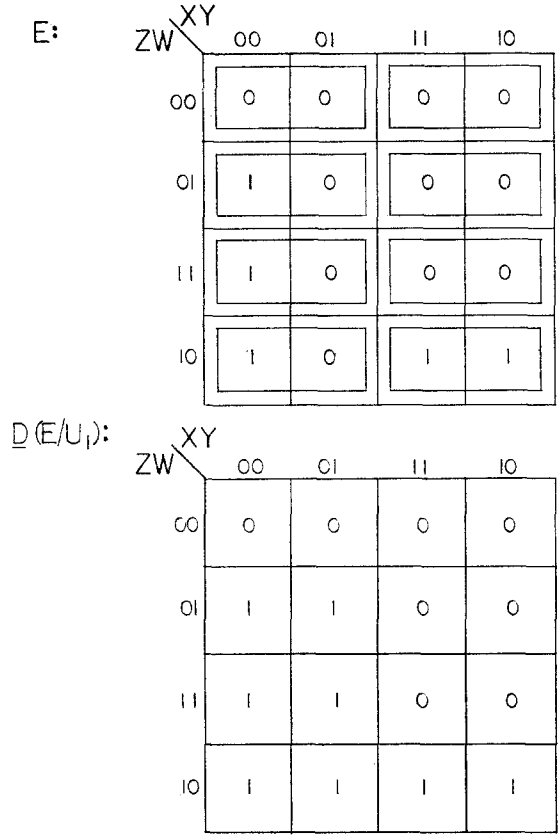

FIG. 3. The effect of the $\underline{D}$-operation.

THEOREM 2. *Given a sum-of-products expression E containing unstable terms, and suppose one of its unstable term is $u_M = U_M S_M$. Then the expression E contains an M variable logic 0 hazard with respect to the variables of U_M within the M-subcubes of U_M not covered by the function H, where $H = \underline{D}(E/U_M) + \bar{S}_M$.*

Proof. The existence of u_M implies that condition (1) as well as condition (2) of Theorem 1 are all covered by $\underline{D}(E/U_M)$. By the definition of acceptable subcubes (Definition 2), the M-subcubes which do not satisfy condition (3) of Theorem 1 are covered by $\bar{S}_M$. Therefore, each uncovered M-subcube of U_M satisfies all the conditions of Theorem 1, and the logic 0 hazard exists in these M-subcubes. Q.E.D.

It is obvious that expression E implies H (i.e., H is always 1 whenever E is 1). Since the value of H does not depend on the variables of U_M, H keeps its value constant during any bridge transition of U_M. By Theorem 2, H is identically 0 during the hazard producing bridge transitions of U_M. Therefore, "AND"ing H with E would eliminate the logic 0 hazard in the M-subcubes not covered by H.

It can be seen that the creation of function H is easy to mechanize and this can serve as the basis of a logic-0 hazard detection algorithm. Further, this same theorem can also be used to detect logic 1 hazard of an expression by inverting it, considering the restrictions of the following theorem.

THEOREM 3. *Given a sum-of-products expression E not containing unstable terms, if we obtain its inverse $\bar{E}$ in sum-of-products form through the use of only the associative, distributive and DeMorgan laws and simplify it by using only the idempotent and absorption laws, then the expression E and $\bar{E}$ interchange* 1 *and* 0 *hazards.*

Proof. It has been proved by Unger [1969] that the above inversion procedure interchanges 1 and 0 hazards. By Lemmas 2 and 3, simplification by idempotent and absorption laws does not change the logic 0 hazard behavior of an expression. Thus the theorem is proved. Q.E.D.

Since for the detection of logic 1 hazard in E we have to apply Theorem 2 to $\bar{E}$, we will be concerned with the inverse of H (denoted by R) in our procedure. Since $\bar{E}$ implies H, R will imply E. Furthermore, R keeps its value 1 during the hazardous bridge transitions of U_M. Therefore, "OR"ing R with E will eliminate the logic 1 hazard within the M-subcubes of U_M not covered by H (i.e., covered by R).

The following theorem (and corollary) shows that inclusion of R in the sum-of-products expression (H in the product-of-sums expression) will not result in any new hazard originally not present in the expression.

THEOREM 4. *Given two sum-of-products expressions E and R. If R "implies" E* (i.e., *E is always* 1 *whenever R is* 1), *then the expression E^* produced by* "OR"*ing E and R will not have any logic* 1 *hazard not present in E.*

Proof. Let us consider two input states I_1 and I_2 where $E^*(I_1) = E^*(I_2) = 1$. If there is no single term in E^* having 1 for both I_1 and I_2 (i.e., the transition from I_1 to I_2 has a 1 hazard), then E can not have such a term too. Thus E^* can not have logic 1 hazards not present in E. Q.E.D.

Obviously the dual of this theorem can be stated as follows:

COROLLARY. *Given two product-of-sums expressions E and H.* If *E implies H* (*i.e., H is always* 1 *whenever E is* 1), *then the expression E* produced by* "AND"*ing E and H will not have any logic* 0 *hazard not present in E.*

After inclusion of all the R's (or H's) corresponding to the unstable terms of $\bar{E}$ (or E), we will have to repeat the whole procedure to see if higher-order logic hazards (hazards involving bigger sets of variables, than already detected), are still there. The repetition of the procedure has to be continued till all the hazards have been detected and eliminated.

Let us show by an example how to use these theorems for logic hazard detection and elimination. Suppose we would like to know whether or not the following expression has logic hazards:

$$E = \bar{z}\bar{w} + \bar{x}y + yw + x\bar{z} + x\bar{y}w.$$

By (1) of Theorem 1, E does not have any logic 0 hazard since it is a sum-of-products expression not containing any unstable term. To detect its logic 1 hazards we take the inverse of E, taking into consideration the restrictions of Theorem 3, resulting in :

$$\bar{E} = \bar{x}\bar{y}z + \bar{x}\bar{y}w + xz\bar{w} + \bar{y}z\bar{w} + y\bar{y}z + x\bar{x}w\bar{w}.$$

Any logic 0 hazard in $\bar{E}$ corresponds to a logic 1 hazard in E within the same subcubes with respect to the same variables. Let us first examine the effect of the unstable term $y\bar{y}z$. This term contains one variable both complemented and uncomplemented. Thus, $u_1 = y\bar{y}z$ has an unstable part $U_1 = y\bar{y}$ and a stable part $S_1 = z$. By Theorem 2, $\bar{E}$ has a logic 0 hazard with respect to the variable y within the 1-subcubes of $y\bar{y}$ not covered by the function $H = \underline{D}(\bar{E}/U_1) + \bar{S}_1 = \bar{x} + \bar{z} + \bar{w}$, i.e., within the 1-subcube 1-11 for the bridge transitions $I_1 \leftrightarrow I_2$ of $y\bar{y}$ where $I_1 = 1011$ and $I_2 = 1111$.

The other unstable term is $u_2 = x\bar{x}w\bar{w}$ with $U_2 = x\bar{x}w\bar{w}$ and $S_2 = 1$. In this case $H = \bar{y} + z$. Thus, $\bar{E}$ has a two variable logic 0 hazard with respect to x and w within the subcube -10-, for the bridge transitions $I_3 \leftrightarrow I_4$ and $I_5 \leftrightarrow I_6$, where $I_3 = 0100$, $I_4 = 1101$, $I_5 = 0101$, and $I_6 = 1100$.

Now we have no more unstable terms in $\bar{E}$. Let us eliminate the hazards already detected. Using Theorem 4 we create the following expression E^* having the same truth table as E, and containing the terms of both $\bar{H}$'s as well as those of E.

$$E^* = \bar{z}\bar{w} + \bar{x}y + yw + x\bar{z} + x\bar{y}w + xzw + y\bar{z}.$$

The inclusion of these terms of $\bar{H}$'s into E^* eliminates the hazards previously

detected. However, E^* still may have some other logic 1 hazards. So let us repeat the whole procedure. Inverting E^* and examining the unstable terms of $\bar{E}^*$ shows that only one of them produces hazard. This unstable term is $U_2 = y\bar{y}z\bar{z}$ with the unstable part $U_2 = y\bar{y}z\bar{z}$ and the stable part $S_2 = 1$. In this case, $H = \bar{x} + \bar{w}$. Thus, the hazardous subcube is 1--1, while the hazardous bridge transitions are $I_7 \leftrightarrow I_8$ and $I_9 \leftrightarrow I_{10}$, where $I_7 = 1001$, $I_8 = 1111$, $I_9 = 1011$, and $I_{10} = 1101$. Including the term $\bar{H} = xw$ into a new expresssion together with E^* and simplifying it by the permissible transformations of Theorem 2, we get $E^{**} = \bar{z}\bar{w} + \bar{x}y + yw + x\bar{z} + y\bar{z} + xw$. Applying the same procedure to E^{**} we find no more logic hazards in E^{**},

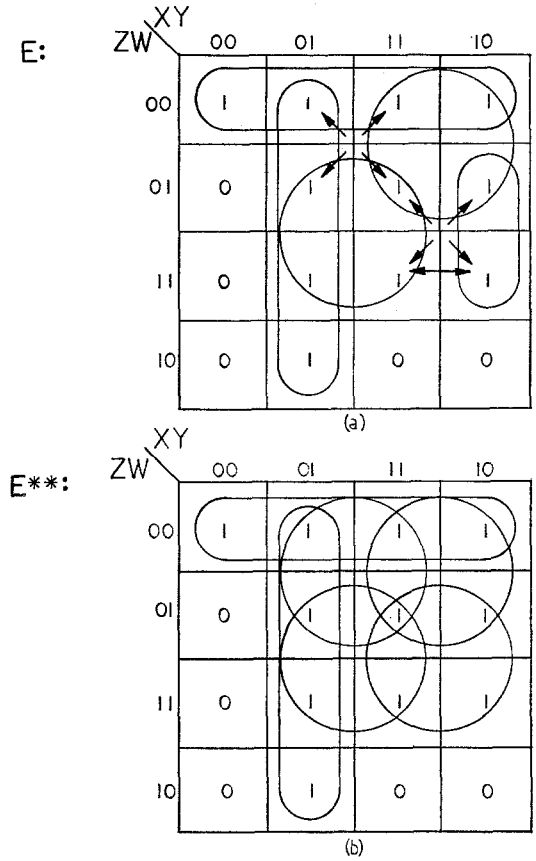

FIG. 4. (a) Karnaugh map of expression E showing transitions with logic 1 hazards. (b) Karnaugh map of expression E^{**}. All the logic 1 hazards of E are eliminated.

and so E^{**} is a logic hazard free expression of E. Figure 4(a) shows the Karnaugh map of E, while Fig. 4(b) shows that of E^{**}. One can see the logic 1 hazards of E, and how they are eliminated in E^{**}.

3. Algorithms for Logic Hazard Detection and Elimination

Two different algorithms have been developed based on the hazard detection and elimination procedure just demonstrated. The first algorithm is intended to save computing time, while the second is for saving memory space. The important parts of the two algorithms are discussed in this section.

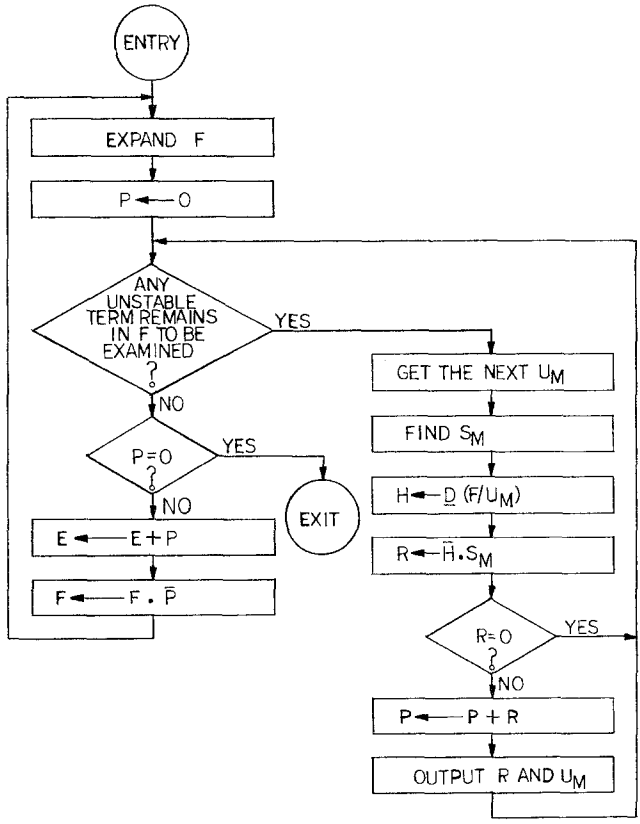

Fig. 5. Flowchart of the Time Saving Algorithm for Logic Hazard Detection and Elimination.

The flowchart of the Time Saving Algorithm for Logic Hazard Detection and Elimination is shown in Fig. 5. The algorithm is concerned with the logic 1 hazards of E, where E is a sum-of-products expression. The expression F is the inverse of E, and is in product-of-sums form. While executing the block "Expand F," the restrictions of Theorem 3 should be taken into consideration. In the step "Get the next U_M," the unstable part of the next unstable term is chosen. Furthermore, in the step "Find S_M," S_M is the sum of the stable parts of all unstable terms having U_M in common.

It is obvious that the steps as well as the organization of this algorithm are very similar to the procedure shown in the previous example. It turns out that this algorithm is also easy to use when one uses pencil and paper.

In Fig. 6 we show the flowchart of the Memory Saving Algorithm for Logic

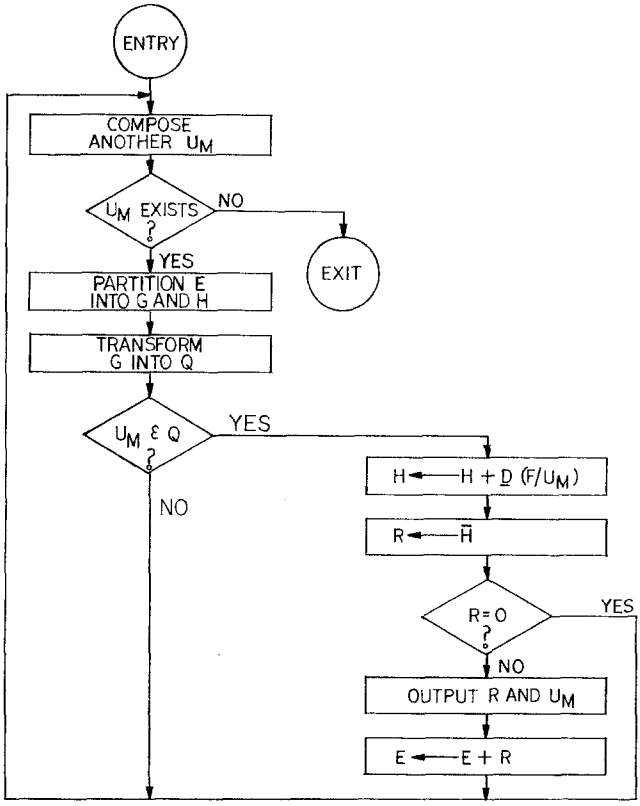

FIG. 6. Flowchart of the Memory Saving Algorithm for Logic Hazard Detection and Elimination.

Hazard Detection and Elimination. In this algorithm, V is the set of variables with respect to which logic 1 hazards of the sum-of-products expression E are to be detected, and V can be defined beforehand. If not defined, V is the whole set of variables in E. The maximum number of variables in V which can change simultaneously (Mm) is also defined initially, and its value, if not defined, is the number of variables in V.

In this algorithm, F is the inverse of E in sum-of-products form with all unstable terms deleted. F need not be directly derived from E. It would suffice if F is *any expression which represents the inverse* function of E.

The block "Compose another U_M" consists of the following four steps:

Step 1. Choose a set of M variables from V ($M = 1$ initially).

Step 2. If all possible combinations of M variables in V have already been chosen, then increase M by 1 and go to Step 3; else go to Step 4.

Step 3. If M is greater than Mm, then exit because U_M does not exit; else go back to Step 1.

Step 4. Create U_M by "AND"ing all variables chosen in Step 1 with their inverses, and then exit.

In the block "Partition E into G and H," each term of E containing any variable of U_M goes to G, the remaining to H. The block "Transform G into Q" means the execution of the following procedure: Get the dual of G in product-of-sums form. Expand it by applying the distribution law successively. After each multiplication, drop out every term which is not able to absorb U_M, then apply the idempotent and absorption laws to the remaining terms. The set of terms in the final result is called Q.

In the time-saving algorithm, all the logic 1 hazards of E are detected in one run. On the other hand, in the memory saving algorithm, only the logic 1 hazards with respect to all possible combinations of the set of variables specified in V are detected. Consequently, the latter algorithm is more flexible as it can use the information given by the designer more effectively. However, when all possible logic 1 hazards are to be detected, this algorithm has to make an exhaustive search and therefore is slower than the time saving one. But the space saving algorithm trades off execution time in favor of less memory space. The same two algorithms, with reinterpretation of the expressions E and F, can be used to detect and eliminate logic 0 hazard.

Both algorithms have been implemented in FORTRAN. The result of one of these programs implementing the time saving algorithm, TSHAZARD, for the earlier example is shown in Fig. 7. The details of the implementations as well as the program listing are given in [Máté *et al.*, 1973).

EXPRESSION TO BE CORRECTED:

$$\neg Z\neg W + \neg XY + WY + \neg ZX + WX\neg Y$$

HAZARD WITHIN THE SUBCUBE: $\neg ZY$

WITH RESPECT TO THE VARIABLES: WX

HAZARD WITHIN THE SUBCUBE: ZWX

WITH RESPECT TO THE VARIABLES: Y

HAZARD WITHIN THE SUBCUBE: WX

WITH RESPECT TO THE VARIABLES: ZY

EXPRESSION CORRECTED:

$$\neg Z\neg W + \neg XY + WY + \neg ZX + \neg ZY + WX$$

FIG. 7. Output of TSHAZARD program, where $\neg A$ denotes $\bar{A}$.

4. A Related Result on Prime Implicant Generation

A related result which is significant by itself is the following theorem. This is similar to one by Nelson [1954] which is often used as the basic principle for prime implicant generation (Slage *et al.*, 1970; Das and Khabra, 1972). We include this theorem here mainly because our proof not only is simple and conceptually easy to understand, but also provides a close relationship between hazard and prime implicant.

THEOREM 5. *Given a sum-of-products expression not containing unstable terms. If we take its inverse* $\bar{E}$ *(or dual* E^D*) through the use of associative, distributive and DeMorgan laws, and simplify it by dropping out the unstable terms and then using the absorption law, then* $\bar{E}$ *(or* E^D*) contains all its prime implicants.*

Proof. Since E does not have any kind of logic 0 hazard (Theorem 1), it follows from Theorem 3 that $\bar{E}$ (or E^D) does not contain any kind of logic 1 hazard.[1] Eichelberger has proved [Eichelberger, 1965] that a sum-of-products realization of a function is logic 1 hazard free *iff* it contains all the prime implicants of the function. But, since $\bar{E}$ (or E^D) is logic 1 hazard free, it has to contain all its prime implicants. Q.E.D.

[1] It is obvious that Theorem 3 remains valid after replacing the inverse $\bar{E}$ with the dual E^D.

5. Conclusions

Logic 0 hazards and terms with complementary literals are closely related. Their relation is analyzed first and then a logic 0 hazard detection procedure is developed based on this relation. A set of logic hazard preserving transformations have been developed and these give a wider generality to the method, making it usable for both logic 0 and logic 1 hazard detection.

Two logic hazard detection and elimination algorithms have been developed using the same basic idea, but with the first one emphasizing the reduction of the execution time and the second one the memory space. Both algorithms are implemented in FORTRAN, although FORTRAN obviously is not efficient for implementing such algorithms which involve mostly symbol manipulation.

Acknowledgments

Mr. Máté is grateful to the Hungarian Academy of Sciences, the National Academy of Sciences (USA) and the Computer Systems Laboratory of Washington University, St. Louis, for supporting his research.

Received: July 30, 1973; revised: July 16, 1974

References

Bredeson, J. and Hulina, P. (1972), Elimination of static and dynamic hazards for multiple input changes in combinational switching circuits, *Inform. Contr.* **20**, 114–124.

Das, Santanu and Chuang, Y. H. (1972), "A New Hazard Detection Technique," Technical Memorandum No. 157, Computer Systems Laboratory, Washington University, St. Louis.

Das, S. and Khabra, N. (1972), Clause-column table approach for generating all the prime implicants of switching functions, *IEEE Trans. Computers* **C-21**, 1239–1246.

Eichelberger, E. (1965), Hazard detection in combinational and sequential switching circuits, *IBM J. Res. Develop.* **9**, 90–99.

Huffman, D. (1957), The design and use of hazard-free switching networks, *J. Assoc. Comput. Mach.* **4**, 47–62.

Máté, L. L., Chuang, Y. H., and Das, S. (1973), "A Logic Hazard Detection and Elimination Method" Technical Memorandum No. 187, Computer Systems Laboratory, Washington University, St. Louis.

McCluskey, E. Jr. (1962), Transients in combinational circuits, *in* "Redundancy Technique for Computing Systems," pp. 9–46, Spartan Book Co., Washington, D.C.

McCluskey, E., Jr. (1956), Minimization of Boolean functions, *Bell System Tech. J.* **35**, 1417–1444.

McGhee, R. (1954), Some aids to the detection of hazards in combinational switching circuits, *IEEE Trans. Computers* **C-18**, 561–565.

Nelson, R. (1954), Simplest normal truth functions, *J. Symbolic Logic* **20**, 105–108.

Rey, C. (1974), Algebra finds logic circuits hazards, *Electr. Design* **22**, 90–92.

Roth, J. P. (1958), Algebraic topological method for the synthesis of switching system I, *Trans. Amer. Math. Soc.* **88**, 301–306.

Slage, J., Chang, C., and Less, R. (1970), A new algorithm for generating prime implicants, *IEEE Trans. Computers* **C-19**, 304–310.

Su, S. H. H. (1969), Automated logic design via a new local extraction algorithm, Proc. Nat. Electron. Conf., 651–656.

Tison, P. (1967), Generalization of consensus theory and application to the minimization of Boolean functions, *IEEE Trans. Electron. Computers* **C-16**, 446–456.

Unger, S. (1969), "Asynchronous Sequential Switching Circuits," pp. 118–138, Wiley–Interscience, New York, 1969.

Yoeli, M. and Rinon, S. (1964), Application of ternary algebra to the study of static hazards, *J. Assoc. Comput. Mach.* **11**, 84–97.